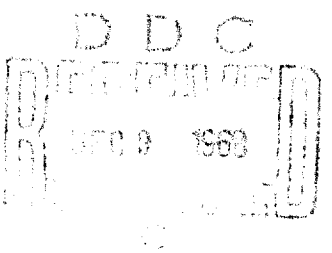
AD 678867

THE SIMSCRIPT [illegible] PROGRAMMING LANGUAGE REFERENCE MANUAL

H. J. Karr and B. [illegible]



PREPARED FOR

[illegible] AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-5778-PR
OCTOBER 1968

THE SIMSCRIPT II PROGRAMMING LANGUAGE: REFERENCE MANUAL

P. J. Kiviat and R. Villanueva

This research is supported by the United States Air Force under Project RAND—Contract No. F44620-67-C-0045—monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
[illegible]

PREFACE AND SUMMARY

SIMSCRIPT II is described completely in P. J. Kiviat, R. Villanueva, and H. M. Markowitz, THE SIMSCRIPT II PROGRAMMING LANGUAGE, The RAND Corporation, R-460-PR, October 1968. This Memorandum, containing only its syntax and semantics, is designed as a reference manual for programmers already familiar with SIMSCRIPT II.

CONTENTS

BLANK PAGE

I. NOTATION

The notation employed in describing SIMSCRIPT II is a combination of conventions used in several computer programming language descriptions. The authors chose it for its convenience in their work and in describing the language to others. In the following pages:

(1) Words in capital letters are statement keywords.

(2) Primitives shown in italics are basic language constructs.

(3) A metavariable denotes an occurrence of an element of the type represented by the metavariable symbol shown in italics.

(4) A statement is a combination of keywords, primitives, and metavariables that follows a certain pattern, called the syntax of the statement. Section III presents the patterns associated with the statements of SIMSCRIPT II and the meanings associated with them, called the semantics of the statements.

(5) Brackets [ ] and braces { } denote choices. When brackets appear, a choice may be made from the options indicated. When braces appear, a choice must be made. The items available for selection appear in a vertical list within the brackets or braces. When a choice can be repeated, a symbol (or symbols) that must separate the items in the list of choices is written at the upper right-hand corner of the brackets or braces. For example, if a choice appears as

$\left\{\begin{matrix}A\\B\end{matrix}\right\}^{,}$ the sequence A,A,B,A, ... ,B might be selected. The choice

represented by $\{A\}^{,}$ is logically equivalent to $-[,A][,A]\ldots[,A]$.

(6) The null character ∞ is used to indicate that no symbol

need separate the items in a list of choices. An example of $\left\{\begin{matrix}A\\B\end{matrix}\right\}^{\text{ONE}}$ might be AABABB... A . The choice represented by $\left\{A\right\}^{\text{ONE}}$ is logically equivalent to A[A][A]...[A] .

(7) A list separator symbol can itself be complex, involving choices and repetitions, as in

$\left\{\begin{matrix}A\\B\end{matrix}\right\}\left\{\begin{matrix}\text{AND}\\\text{OR}\end{matrix}\right\}$ an instance of which might be A AND B OR A OR B .

(8) Plural keywords ending in S such as VARIABLES and LINES , can be written in singular form as VARIABLE or LINE when called for by the grammar of a statement.

II. BASIC CONSTRUCTS

SYMBOLS

character = { *letter*, *special character*, *digit*, *blank*, *period* }

letter = { A B C D E F G H I J K L M N O P Q R S T U V W X Y Z }

special character = { + - * / = ( ) $, ' }

digit = { 0 1 2 3 4 5 6 7 8 9 }

blank = empty space

period = .

PRIMITIVES

name = $\left\{ \begin{matrix} letter \\ digit \\ period \end{matrix} \right\}^{\infty}$	A name (a) must contain at least one letter or two periods (b) does not terminate with one or more periods
integer i = $\{digit\}^{\infty}$	
number n = $\left\{ \begin{matrix} digit \\ period \end{matrix} \right\}^{\infty}$	A number contains at most one period and at least one digit
string s = "$[character]^{\infty}$"	$[character]^{\infty}$ cannot contain a "
text t = |$[character]^{\infty}$|	$[character]^{\infty}$ cannot contain a |

METAVARIABLES

$$\text{word } w = \left\{ \begin{array}{l} i \\ \text{name} \\ n \\ \textit{special character} \\ s \end{array} \right\}$$

Words must be separated from each other by one or more blanks unless they are special characters.

Periods (.) are ignored between words and at the end of statements.

Comments can be inserted between any two words in a program by enclosing them in quote marks ('') formed by two consecutive apostrophes. The right-hand set of quotes is not necessary if the comment is the last item on a card.

$$\text{routine } r = \text{'name'}$$

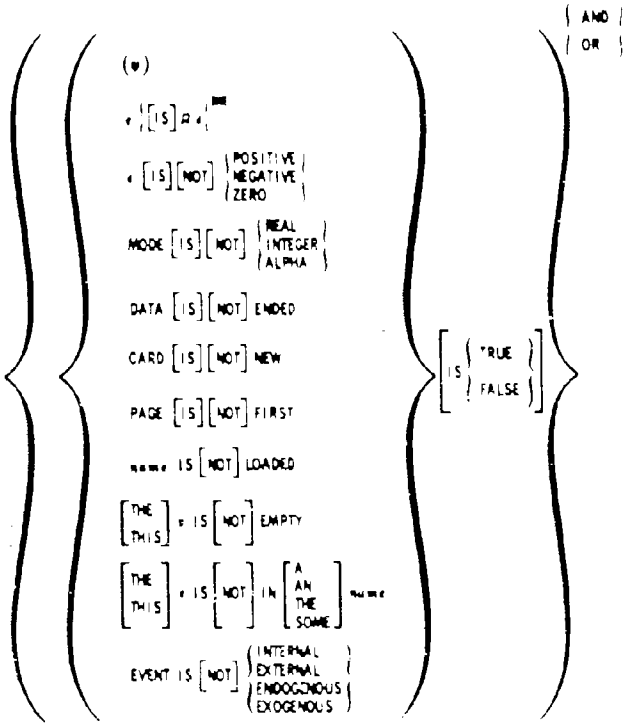
$$\text{comma } c = \left\{ \begin{array}{l} , \\ \text{AND} \\ \text{AMD} \end{array} \right\}$$

$$\text{label } l = \left\{ \begin{array}{l} \text{name} \\ n \end{array} \right\}$$

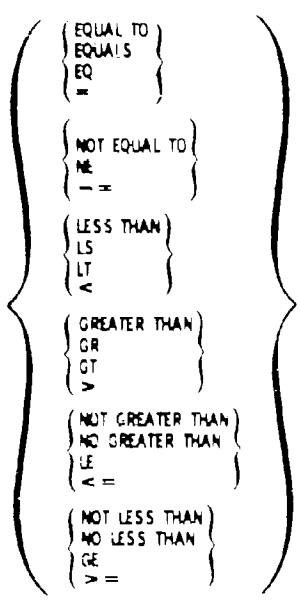
$$\text{variable } v = \text{name} \left[\left(\left\{ \begin{array}{l} (e) \left[, (e)' \right] \\ (e)' \end{array} \right\} \right) \right]$$

$$\text{arithmetic expression } e = \left[\begin{array}{l} + \\ - \end{array} \right] \left\{ \begin{array}{l} (e) \\ n \\ r \\ s \\ l \\ [s]v \end{array} \right\} \left\{ \begin{array}{l} + \\ - \\ * \\ / \end{array} \right\}$$

logical expression ¢ =



relational operator ρ =



termination clause tc = $\left\{ \begin{matrix} \text{WHILE} \\ \text{UNTIL} \end{matrix} \right\} \ddagger [\cdot]$

selection clause sc = $\left\{ \begin{matrix} \text{WITH} \\ [\text{EXCEPT}]\ \text{WHEN} \\ \text{UNLESS} \end{matrix} \right\} \ddagger [\cdot]$

format f_1 = $\left\{ \begin{matrix} \text{B}\,e \\ \text{S}\,e \\ / \end{matrix} \right\}$

format f_2 = $\left\{ \begin{matrix} f \\ \text{C}\,e \\ \text{I}\,e \\ \text{A}\,e \\ \text{D}\,e,e \\ \text{E}\,e,e \\ \text{T}\,e \end{matrix} \right\}$

for phrase for =

$$\text{FOR} \left\{ \begin{matrix} \text{name} \left\{ \begin{matrix} \text{BACK FROM} \\ = \end{matrix} \right\} e \ \text{TO}\ e \left[\text{BY}\ e \right] \\ \left\{ \begin{matrix} \text{EACH} \\ \text{EVERY} \\ \text{ALL} \end{matrix} \right\} \left\{ \begin{matrix} \text{name} \left[\text{CALLED name} \right] \\ \text{name} \left[\left\{ \begin{matrix} \text{FROM} \\ \text{AFTER} \end{matrix} \right\} e \right] \left\{ \begin{matrix} \text{OF} \\ \text{IN} \\ \text{ON} \\ \text{AT} \end{matrix} \right\} e \left[\text{IN REVERSE ORDER} \right] \end{matrix} \right\} \\ \text{EACH name IN THE DICTIONARY} \end{matrix} \right\} \left[\cdot \right] \left[\begin{matrix} wc \\ tc \end{matrix} \right]^{\blacksquare}$$

III. STATEMENTS

NONEXECUTABLE

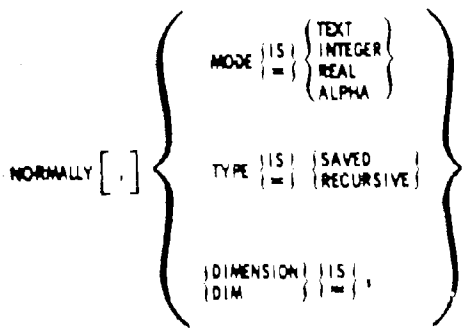
(1) PREAMBLE

Marks the beginning of the program preamble.

(2) LAST COLUMN { IS / = } i

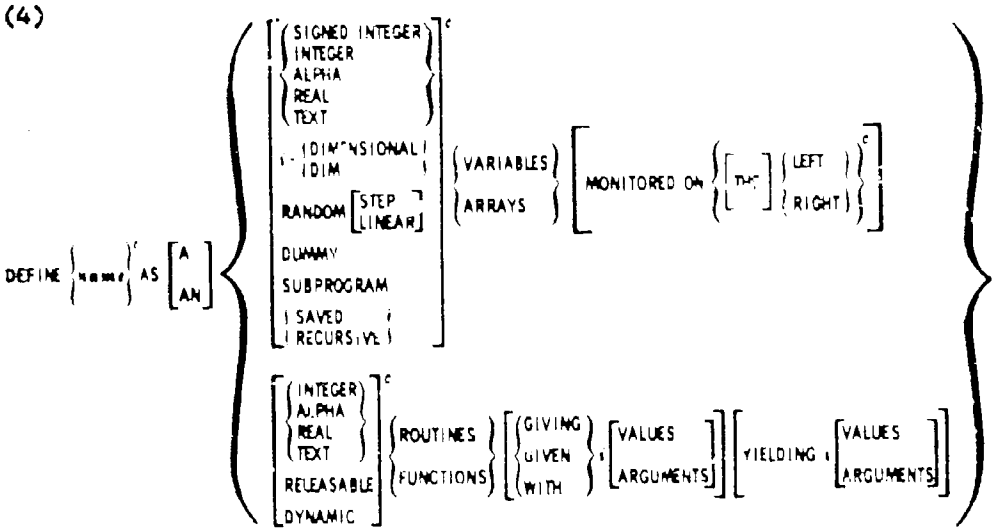
Characters beyond column i are ignored on subsequent cards.

(3)



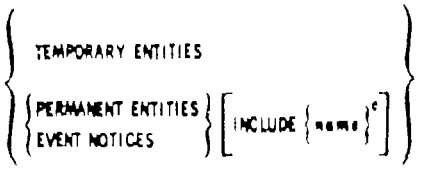
Establishes background conditions for properties of variables and functions that are effective unless overridden by subsequent DEFINE declarations or, in the case of local arrays, first use.

(4)



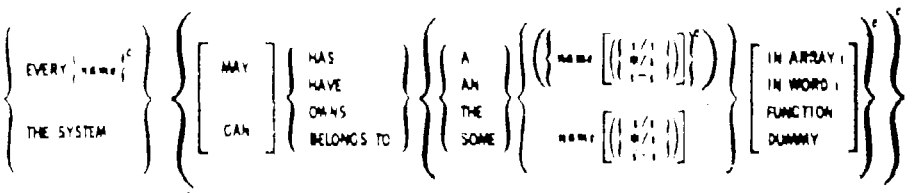
Defines properties of global and local variables, and routines.

(5)



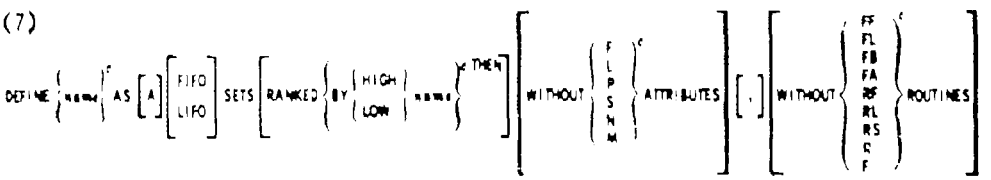
Declares the type of following EVERY statements.

(6)



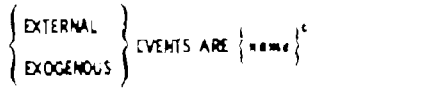
Entity-attribute-set structure declaration. Specifies attribute packing, equivalence, word assignment and function options.

(7)



Defines set ranking, owner and member attributes and generated set processing routines.

(8)

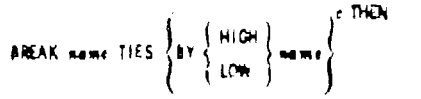


Declares the names of events that can be triggered externally.

(9)

Names units from which external event data will be read.

(10)

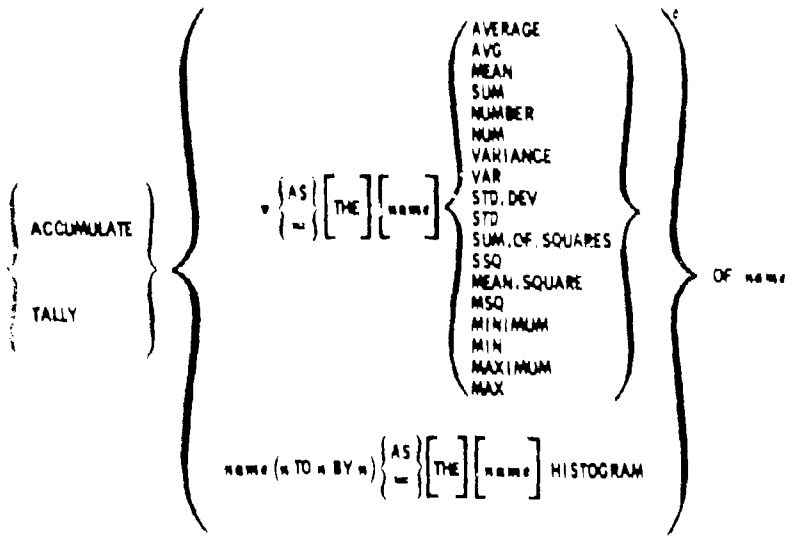


Establishes a priority order within an event class.

(11) PRIORITY ORDER IS {name}^c

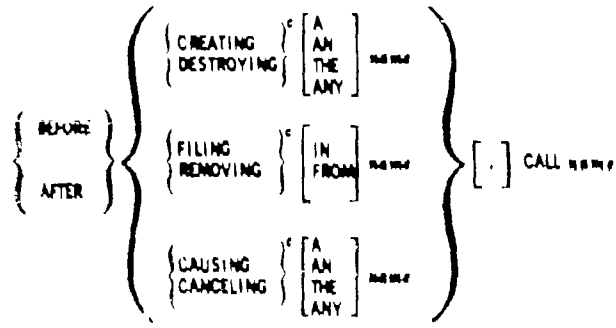
Assigns a priority order to different classes of events.

(12)



Specifies automatic data collection and analysis for named variables.

(13)



Specifies a call to a named routine whenever the indicated statement is executed. Inputs to the routines are:

	BEFORE	AFTER
CREATE	Not allowed	Entity identifier
DESTROY	Entity identifier	Not allowed
CAUSE	Entity identifier, time	Entity identifier, time
CANCEL	Entity identifier	Entity identifier
FILE	Entity identifier, subscripts	Entity identifier, subscripts
REMOVE	Entity identifier, subscripts	Entity identifier, subscripts

(14) DEFINE w TO MEAN $\left\{ w \right\}^{m}$

Instructs the compiler to substitute the words (up to the end of the card on which the statement appears) following the keyword MEAN for the indicated word in all subsequent statements, before they are compiled.

(15) SUBSTITUTE $\left\{ \begin{matrix} \text{THIS} \\ \text{THESE} \end{matrix} \right\}$, LINES FOR w

Similar to (14) but allows more than one card of words to be substituted.

(16) $\left\{ \begin{matrix} \text{SUPPRESS} \\ \text{RESUME} \end{matrix} \right\}$ SUBSTITUTION

Used to override currently defined substitutions. These statements must not be placed on program cards with other statements.

(17) END

Marks the end of a program preamble, routine, report section, and heading block of a report section.

(18) MAIN

Marks the beginning of a program's main routine. Execution commences at the first executable statement after MAIN.

(19) $\left[\begin{matrix} \text{LEFT} \\ \text{RIGHT} \end{matrix} \right]$ ROUTINE $\left[\begin{matrix} \text{TO} \\ \text{FOR} \end{matrix} \right]$ name $\left[\left\{ \begin{matrix} \text{THE} \\ \text{THIS} \\ \text{GIVING} \\ \text{GIVEN} \end{matrix} \right\} \left\{ \text{name} \right\}^{c} \right] \left[\left(\left\{ \text{name} \right\}^{c} \right) \right]$ $\left[\text{YIELDING} \left\{ \text{name} \right\}^{c} \right]$

Subprogram declaration. Routines used as functions only have GIVEN arguments. If LEFT or RIGHT are not stated, RIGHT is implied.

(20) $\left\{\begin{matrix}\text{UPON}\\\text{EVENT}\end{matrix}\right\}$ name $\left[\left\{\begin{matrix}\text{THE}\\\text{THIS}\\\text{GIVEN}\\\text{GIVING}\end{matrix}\right\}\text{name}\left\{,\right\}\right]\left[\left(\text{name}\left\{,\right\}\right)\right]\left[\text{SAVING THE EVENT NOTICE}\right]$

Event declaration. Unless SAVED , an event notice is destroyed before an event routine is executed.

(21) BEGIN REPORT [ON A NEW PAGE] [PRINTING *or* IN GROUPS OF 1 [PER PAGE]]

Marks the beginning of a report section.

(22) BEGIN HEADING

Marks the beginning of a heading block within a report section.

A PREAMBLE STATEMENT RECAP

Statement Type	Statement	Rules
1a	NORMALLY	Can appear anywhere in preamble.
1b	DEFINE TO MEAN	
1c	SUBSTITUTE	
1d	SUPPRESS SUBST	
1e	RESUME SUBST	
2a	TEMPORARY ENTITIES	A preamble may contain many Type 2a, 2b, and 2c statements. Each may be followed by a group of Type 3a, 4, and 5 statements.
2b	PERMANENT ENTITIES	
2c	EVENT NOTICES	
3a	EVERY	Many can follow a Type 2 statement. An entity or event notice name can appear in more than one EVERY statement.
3b	THE SYSTEM	
4	DEFINE VARIABLE	No precedence relation if it defines a global variable. Must follow all Type 3a statements if it defines an attribute named in them A variable, attribute, or function name can appear in only one DEFINE statement.
5	DEFINE SET	Must follow Type 4 statements in a Type 2 statement group if it qualifies a set named in them.
6a	BREAK TIES	One statement allowed for each event notice.
6b	EXTERNAL EVENTS	
6c	EXTERNAL UNITS	
7	PRIORITY	Must follow all Type 2c and 6b statements.
8a	BEFORE	Allowed for each temporary entity, set, and event notice.
8b	AFTER	
9a	TALLY	One statement allowed for each global variable or attribute.
9b	ACCUMULATE	

Of these statements, only Types 1 and 4 can be used in routines to declare local background conditions, variables, and substitutions.

STORAGE ALLOCATION

(1) RESERVE { v }^c AS { e }^BY [BY e]^c

Allocates blocks of core of specified size to the pointer variables v. Words assigned are data if no BY e phrase appears, and are pointers otherwise.

(2) RELEASE { v }^c

Releases blocks of core pointed to by v; v's are assumed to be pointer variables.

(3) CREATE { [A / AN] name [CALLED v] | { EACH / ALL / EVERY } name [(e)]^c }

Obtains a block of words from the "free-storage" area.

(4) DESTROY [THE / THIS] name [CALLED v]

Returns a specified block of words to the "free-storage" area.

(5) ERASE { v }^c

Removes text variables from the dictionary and returns their space to the "free-storage" area.

COMPUTATION

(1) LET v = e

Assigns the value of e to the variable v.

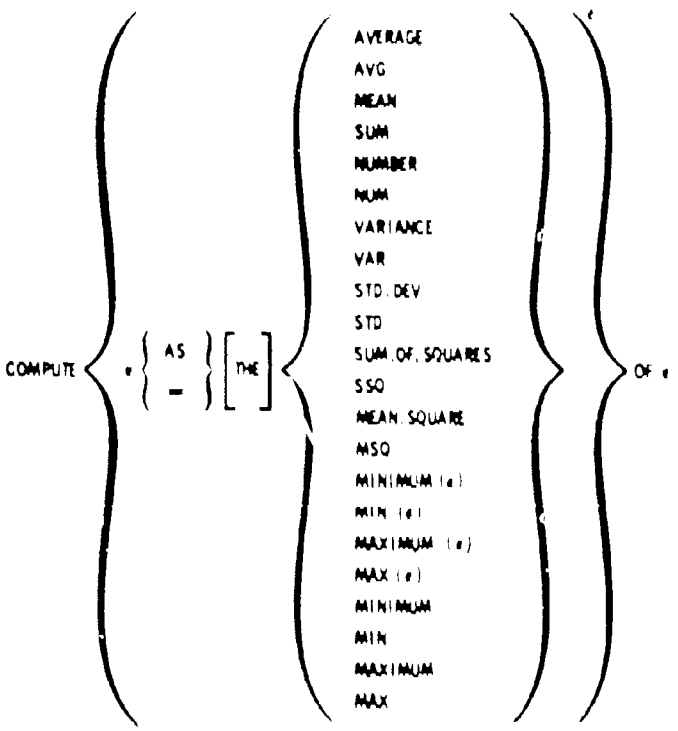
(2) ADD e TO v

Adds the value of e to the value of the variable v.

(3) SUBTRACT e FROM v

Subtracts the value of e from the value of the variable v.

(4)



Must be controlled by a logical control phrase. Computes the indicated statistics of the expression e after the LOOP statement if the control is over a DO... LOOP block.

(5) FIND { THE FIRST CASE / { v = [THE][FIRST], v }^c } [.] [IF { FOUND / NONE } [.]]

Must be controlled by a logical control phrase, but cannot be within a DO... LOOP block. The optional IF phrase directs control after the control phrase has been completed, depending upon the "success" of the FIND.

(6) FILE [THE / THIS], [FIRST / LAST / { BEFORE / AFTER }, v] IN [THE / THIS], v

Files an entity in a set.

(7) REMOVE [THE] { { FIRST / LAST }, v / [THIS / ABOVE], } FROM [THE / THIS], v

Removes an entity from a set.

(8) MOVE { FROM v / TO v }

Used within a routine defined for a monitored variable to access or set the value of the variable.

(9) ENTER WITH v

Used to transfer a "right-hand" value to a left-handed function.

(10) STORE v IN v

Assigns a value to a variable without mode conversion.

(11) RESET [name] TOTALS OF { v }^c

Initializes ACCUMULATE or TALLY counters associated with v. If TOTALS is not qualified by a word, all counters of v are initialized.

<u>CONTROL</u>

(1) $l\left[(e)\right]$

A statement label identifies a transfer point.

(2) $$\text{GO}\left[\text{TO}\right]\left\{\begin{matrix} l\left[(e)\right] \\ i\left[(e)\right] \end{matrix}\right\}$$

Transfers control to the indicated label.

(3) $$\text{GO}\left[\text{TO}\right]\left\{\begin{matrix} l \\ l' \end{matrix}\right\}^{\left\{\begin{matrix} , \\ \text{OR} \end{matrix}\right\}} \text{PER } e$$

Transfers control to the n^{th} label in the label list according to the integer value of the transfer expression e.

(4) $$\left[\text{THEN}\right] \text{IF } e \left[\,,\,\right]$$

If the logical expression e is true, continues execution with the next statement. If e is false, transfers to the following ELSE statement. When nested IF statements appear, the word THEN can be used to indicate that they have a common ELSE statement.

(5) $$\left\{\begin{matrix} \text{ELSE} \\ \text{ALWAYS} \\ \text{OTHERWISE} \\ \text{REGARDLESS} \end{matrix}\right\}$$

Synonyms indicating the transfer point of the false condition of a preceding IF statement.

(6) $$\left[\text{ALSO}\right]\left\{\begin{matrix} for \\ ic \end{matrix}\right\}\left[\begin{matrix} for \\ ic \\ ac \end{matrix}\right]^{*} \text{DO}\left[\begin{matrix} \text{THIS} \\ \text{THE FOLLOWING} \end{matrix}\right]$$

Logical phrases control the execution of statements that follow them. When more than one statement is to be controlled, the

word DO precedes them. Multiple control phrases terminating control on the same LOOP statement are preceded by the word ALSO.

(7) { LOOP / REPEAT }

Used with DO to delimit a group of statements controlled by one or more logical control phrases.

(8) { PERFORM / CALL / NOW } name [[{ THE / THIS / GIVEN / GIVING }] { , }^c] [YIELDING { , }^c]

Calls a routine used as a procedure. Both input GIVEN and output YIELDING argument lists are optional.

(9) RETURN [{ ( ) / WITH e }]

Used as a procedure, a routine returns control to its calling program with the statement RETURN ; used as a function, a routine returns control and a value to its calling program by either of the statements RETURN (e) or RETURN WITH e.

(10) STOP

Halts program execution.

INPUT-OUTPUT

(1)

PRINT i [DOUBLE] LINES [WITH { , / A GROUP OF e {,/ } FIELDS }^c] [SUPPRESSING FROM COLUMN i] { LIKE THIS / THUS / AS FOLLOWS }

The i lines following the PRINT statement are format lines containing text and pictorial formats for the display of indicated expression values. The phrases A GROUP OF e {,/ } FIELDS and SUPPRESSING FROM COLUMN i can only be used within report sections that have column repetition.

(2)

LIST { , / ATTRIBUTES OF { name [CALLED e] / EACH name [OF r] } }^c

A free-form output statement that labels and displays values of expressions and 1- and 2-dimensional arrays.

(3)

USE { THE BUFFER / [TAPE / UNIT] e } FOR { INPUT / OUTPUT }

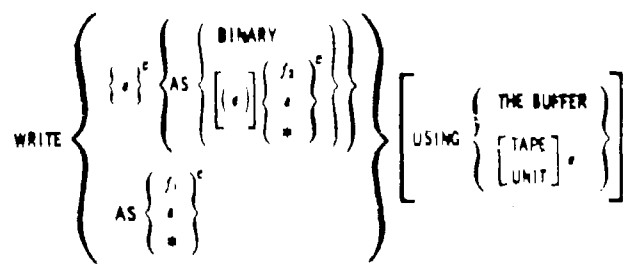
Sets the indicated input/output device as the current input or output unit. All subsequent input/output statements that do not specify their own devices in USING phrases use these current units. THE BUFFER causes reading and writing in an internal file.

(4)

READ { i {,/ }^c [AS { BINARY / [e] i {,/ }^c }] / AS i {,/ }^c } [USING { THE BUFFER / [TAPE / UNIT] e }]

Used without an AS clause indicates a free-form data input.

(5)



Writes formatted output only.

(6)

Rewinds an input/output device.

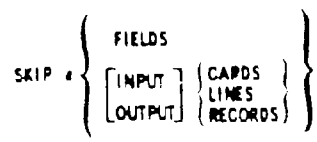
(7)

Performs the indicated operations.

(8)

Writes an end-of-file mark on an output device.

(9)



Applies to the current input or current output unit. SKIP * FIELDS applies to the current input unit only when it is used for free-form data input. CARDS, LINES, and RECORDS are synonyms. If neither INPUT nor OUTPUT is specified, INPUT is implied.

(10) START NEW { PAGE | [INPUT / OUTPUT] { LINE / CARD / RECORD } }

Applies to the current input or current output unit. LINE, CARD, and RECORD are synonyms. If neither INPUT nor OUTPUT is specified, INPUT is implied.

(11) INPUT { v / f } [USING [TAPE / UNIT] e]

Reads data as successive characters in a TEXT variable until the character contained in MARK.V is encountered.

(12) OUTPUT { v / a / f / t / * } [USING [TAPE / UNIT] e]

Writes TEXT variables, starting at the current output column.

(13) TRACE [USING [TAPE / UNIT] e]

Produces a backtrack of current subprogram calls. When the SIMSCRIPT II operating system uses TRACE the standard output device (printer) is used.

(14) LOAD name

Used either with program overlay or dynamic program relocation to load an indicated routine from a system load unit.

(15) SAVE {name}

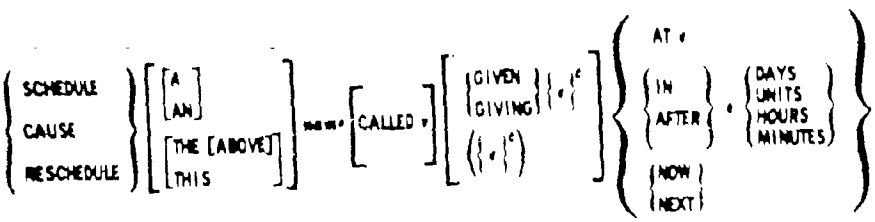
Used only with dynamic program relocation. Saves an indicated routine on a system save unit.

SIMULATION

(1) START SIMULATION

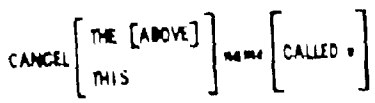
Starts simulation by removing the first event from the events set and executing it.

(2)



Files an event notice in the events set according to its time.

(3)



Removes a scheduled event notice from the event set.

IV, SYSTEM-DEFINED VALUES

CONSTANTS

EXP.C	e
INF.C	Largest INTEGER value that the computer can store
PI.C	π
RADIAN.C	57.29577 degrees/radian
RINF.C	Largest REAL value that the computer can store

VARIABLES

Name	Definition	Default Value
BETWEEN.V	SUBPROGRAM variable called before events are executed	0
BUFFER.V	Length of THE BUFFER	Implementation
EOF.V	End-of-file action code	0
EVENT.V	Code of the current event	0
EVENTS.V	Number of event classes	0
F.EV.S	Array containing first-in-set pointers for set EV.S	0
HOURS.V	Number of hours per simulated day	24
LINE.V	Number of current output line	0
LINES.V	Number of lines allowed per page	55
MARK.V	TEXT, external event data and RANDOM variable read termination character	"*"
MINUTES.V	Number of minutes per simulated hour	60
PAGE.V	Number of current page	1
RCOLUMN.V	Location in buffer of current read pointer	0
READ.V	Number of current read unit	Implementation
SEED.V	Array containing initial random numbers	Implementation
TIME.V	Current simulation time	0
WCOLUMN.V	Location in buffer of current write pointer	0
WRITE.V	Number of current write unit	Implementation

V. SYSTEM-DEFINED ROUTINES

ORIGIN.R(e_1, e_2, e_3) Establishes a simulation-time origin

e_1, e_2, e_3 INTEGER

VI. GENERATED ATTRIBUTES, VARIABLES AND ROUTINES

SETS	Generated attributes	P.set, S.set, M.set } Member F.set, L.set, N.set } Owner
	Generated routines	A.set File first or ranked B.set File last C.set File before D.set File after X.set Remove first Y.set Remove last Z.set Remove specific
ENTITIES	Generated variables	N.entity W.entity E.entity
	Generated routine	C.entity
EVENT NOTICES	Generated variables	T.entity W.entity E.entity
	Generated routines	C.entity
STANDARD ENTITIES	Generated attributes	RANDOM.E: PROB.A, IVALUE.A, RVALUE.A EVENT NOTICE: TIME.A, EUNIT.A, P.EV.S, S.EV.S, M.EV.S TEXT VARIABLE: LENGTH.A
ACCUMULATE VARIABLES	Generated routine	R.variable

VII. LIBRARY FUNCTIONS

Name	Arguments	Operation	Mode	Restrictions
ABS.F	e	$\|e\|$	Mode of e	None
ARCCOS.F	e	arccos(e)	REAL	$-1 \le e \le 1$ and REAL
ARCSIN.F	e	arcsin(e)	REAL	$-1 \le e \le 1$ and REAL
ARCTAN.F	e_1, e_2	arctan(e_1/e_2)	REAL	$(e_1, e_2) \neq (0.0)$ and REAL
ATOT.F	v	Converts the characters of the ALPHA variable v to TEXT	TEXT	v a variable and ALPHA
BINOMIAL.F	e_1, e_2, e_3	Random sample from a binomial distribution with number of trials = e_1, probability of success = e_2 using random number stream e_3	INTEGER	e_1, e_3 INTEGER; e_2 REAL
CONCAT.F	v_1, v_2	Concatenation of v_1 and v_2	TEXT	v_1, v_2 TEXT
COS.F	e	cos(e)	REAL	e in radians and REAL
DATE.F	e_1, e_2, e_3	Converts month, day, year to cumulative time	REAL	e_1, e_2, e_3 INTEGER
DAY.F	e	"day part" of time expression e	INTEGER	e REAL
DIM.F	v	Number of elements pointed to by v	INTEGER	v a pointer variable
DIV.F	e_1, e_2	TRUNC.F(e_1/e_2)	INTEGER	e_1, e_2 INTEGER; $e_2 \neq 0$
EFIELD.F	——	Ending column of next field to be read in free-form data input	INTEGER	——
ERLANG.F	e_1, e_2, e_3	Random sample from Erlang distribution with mean = e_1, $k = e_2$ using random number stream e_3	REAL	e_1 REAL; e_2, e_3 INTEGER
EXP.F	e	exp(e)	REAL	e REAL
EXPONENTIAL.F	e_1, e_2	Random sample from exponential distribution with mean = e_1 using random number stream e_2	REAL	e_1 REAL; e_2 INTEGER
FRAC.F	e	e - TRUNC.F(e)	REAL	e REAL
GAMMA.F	e_1, e_2, e_3	Random sample from a Gamma distribution with mean = e_1, $k = e_2$ using random number stream e_3	REAL	e_1, e_2 REAL; e_3 INTEGER
HOUR.F	e	"hour part" of time expression e	INTEGER	e REAL
INT.F	e	e rounded to an integer	INTEGER	e REAL
ISTEP.F	v, e	Random sample from a look-up table pointed to by v using random number stream e	INTEGER	v a pointer variable; e INTEGER
ITOA.F	e	Converts the integer expression e to ALPHA	ALPHA	e INTEGER
LIN.F	v, e	Random sample from a look-up table pointed to by v using random number stream e applying linear interpolation	REAL	v a pointer variable, e INTEGER
LOG.E.F	e	ln(e)	REAL	$e > 0$ and REAL
LOG.NORMAL.F	e_1, e_2, e_3	Random sample from a lognormal distribution with mean = e_1, standard deviation = e_2 using random number stream e_3	REAL	e_1, e_2 REAL; e_3 INTEGER
LOG.10.F	e	log(e)	REAL	$e > 0$ and REAL

VII. LIBRARY FUNCTIONS *(continued)*

Name	Arguments	Operation	Mode	Restrictions
MAX.F	$e_1, \ldots, e_n$	Value of largest e_i	INTEGER if all e_i INTEGER, otherwise REAL	None
MIN.F	$e_1, \ldots, e_n$	Value of smallest e_i	INTEGER if all e_i INTEGER, otherwise REAL	None
MINUTE.F	e	'minute part' of time expression e	INTEGER	e REAL
MOD.F	e_1, e_2	e_1-TRUNC.F(e_1/e_2)*e_2	INTEGER if e_1 and e_2 INTEGER, otherwise REAL	$e_2 \neq 0$
MONTH.F	e	'month part' of time expression e	INTEGER	e REAL
NORMAL.F	e_1, e_2, e_3	Random sample from a normal distribution with mean = e_1, standard deviation = e_2, using random number stream e_3	REAL	e_1, e_2 REAL; e_3 INTEGER
OUT.F	e	ALPHA value of e^{th} character in the current output buffer	ALPHA	$e \geq 0$ and INTEGER
POISSON.F	e_1, e_2	Random sample from a Poisson distribution with mean = e_1 using random number stream e_2	INTEGER	e_1 REAL, e_2 INTEGER
RANDOM.F	e	Pseudorandom number on interval (0,1) using random number stream e	REAL	e INTEGER
REAL.F	e	e expressed as a REAL number	REAL	e INTEGER
RSTEP.F	t, e	Random sample from look-up table pointed to by t using random number stream e	REAL	t a pointer variable, e INTEGER
SFIELD.F	——	Starting column of next field to be read in free-form data input	INTEGER	——
SIGN.F	e	1 if $e > 0$ 0 if $e = 0$ -1 if $e < 0$	INTEGER	e REAL
SIN.F	e	$\sin(e)$	REAL	e in radians and REAL
SQRT.F	e	$\sqrt{e}$	REAL	$e \geq 0$ and REAL
TAN.F	e	$(\sin(e)/\cos(e))$	REAL	e in radians and REAL
TRUNC.F	e	e-FRAC.F(e)	INTEGER	e REAL
TTOA.F	t	Converts the initial characters of the TEXT string t to ALPHA	ALPHA	t a TEXT variable
UNIFORM.F	e_1, e_2, e_3	Random sample from a uniform distribution over interval (e_1, e_2) using random number stream e_3	REAL	e_1, e_2 REAL, e_3 INTEGER
WEEKDAY.F	e	'day of the week' of time expression e	INTEGER	e REAL
WEIBULL.F	e_1, e_2, e_3	Random sample from a Weibull distribution with scale parameter = e_1 and shape parameter = e_2 using random number stream e_3	REAL	e_1, e_2 REAL, e_3 INTEGER
YEAR.F	e	'year part' of time expression e	INTEGER	e REAL

DOCUMENT CONTROL DATA

1. ORIGINATING ACTIVITY	2a. REPORT SECURITY CLASSIFICATION
THE RAND CORPORATION	UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

THE SIMSCRIPT II PROGRAMMING LANGUAGE: REFERENCE MANUAL

4. AUTHOR(S) (Last name, first name, initial)

Kiviat, P. J. and R. Villanueva

5. REPORT DATE	6a. TOTAL No. OF PAGES	6b. No. OF REFS.
October 1968	32	---
7. CONTRACT OR GRANT No.	8. ORIGINATOR'S REPORT No.	
F44620-67-C-0045	RM-5776-PR	
9a. AVAILABILITY / LIMITATION NOTICES	9b. SPONSORING AGENCY	
DDC-1	United States Air Force Project RAND	

10. ABSTRACT

A compact reference listing of the syntax and semantics of SIMSCRIPT II, designed for professional programmers already familiar with the language. (SIMSCRIPT II is fully described in R-460-PR, and its IBM 360 implementation in RM-5777-PR.) The notation employed was chosen for convenience and descriptive power from conventions previously used in computer programming language descriptions. The study describes notation; basic constructs (symbols, primitives, metavariables); statements (non-executable, storage allocation, computation, control, input-output, simulation); system-defined values (constants, variables); a system-defined routine (the ORIGIN routine for simulation time); generated attributes, variables, and routines; and library functions.

11. KEY WORDS

SIMSCRIPT (Programming Language)
Computer Programming Language